

Defined Architecture of ANN Model

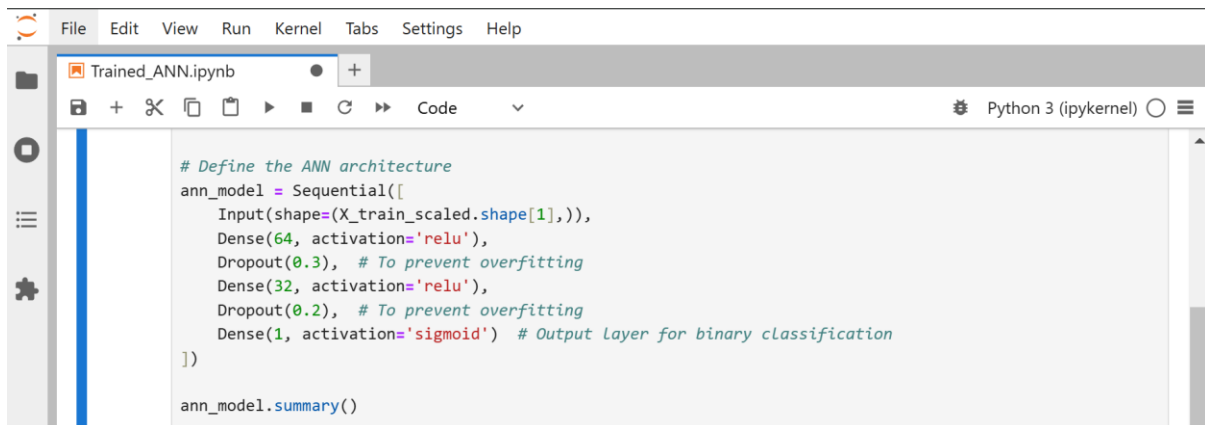
Defining the architecture of an Artificial Neural Network (ANN) involves determining:

- Number of layers
- Type of layers
- Number of neurons in each layer
- Activation functions

Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Input
```

Import all necessary libraries into project



The screenshot shows a Jupyter Notebook window titled 'Trained_ANN.ipynb'. The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar with icons for file operations and execution. The code cell contains the following Python code:

```
# Define the ANN architecture
ann_model = Sequential([
    Input(shape=(X_train_scaled.shape[1],)),
    Dense(64, activation='relu'),
    Dropout(0.3), # To prevent overfitting
    Dense(32, activation='relu'),
    Dropout(0.2), # To prevent overfitting
    Dense(1, activation='sigmoid') # Output Layer for binary classification
])

ann_model.summary()
```

Explanation

Model type: Sequential

```
ann_model = Sequential([
```

The Sequential model is chosen for this project because it aligns well with the requirements of a binary classification problem like predicting customer churn.

The Sequential model in Keras allows us to stack layers in a linear order, where the output of one layer is the input to the next. This approach is suitable for simple feedforward neural networks.

Input layer

```
Input(shape=(X_train_scaled.shape[1],)),
```

Defines the input shape for the model.

X_train_scaled.shape[1]: Represents the number of features (columns) in the training dataset.

- In the **X_train_scaled** dataset, there are 10 features (columns). So, the input shape becomes **(10,)**, meaning the model expects input vectors of size 10

This layer does not have trainable parameters. It is only for defining the structure of the model.

The trailing comma makes it a tuple, which is required by the shape argument. Without the comma, it would not be interpreted as a tuple but as an integer (which is invalid for shape).

First Dense layer (Hidden layer)

```
Dense(64, activation='relu'),
```

A Dense (fully connected) layer with 64 neurons.

Activation function:

- relu (Rectified Linear Unit):
 - Introduces non-linearity into the model.
 - Outputs:
 - 0 for negative inputs
 - The input value itself for positive inputs.

Purpose:

- The layer learns 64 features by transforming the input data through weights and biases.
- This layer learns a broad set of features from the input data.

- Having more neurons allows the model to capture a wide range of patterns and relationships.

Dropout layer (after first Dense layer)

```
Dropout(0.3), # To prevent overfitting
```

Dropout is a regularization technique to prevent overfitting.

0.3 means 30% of the neurons in previous layer (64 neurons) are randomly set to 0 during training (at each iteration).

This forces the network to learn more robust patterns rather than relying on specific neurons.

Second Dense layer (Hidden layer)

```
Dense(32, activation='relu'),
```

A second Dense layer with 32 neurons.

Use the same ReLU activation as the first Dense layer. Further transforms the features learned by the first Dense layer into a more compact representation.

Purpose:

- Refine and distill the features learned by earlier layer.
- Fewer neurons help focus on the most important patterns, reducing redundancy.

Dropout layer (after second Dense layer)

```
Dropout(0.2), # To prevent overfitting
```

0.2 means 20% of the neurons in the previous layer (32 neurons) are randomly deactivated during training. Provides additional regularization to prevent overfitting.

Output Layer

```
Dense(1, activation='sigmoid') # Output layer for binary classification
```

A Dense layer with 1 neuron because this is a binary classification problem (churn or no churn).

Activation function:

- sigmoid:
 - It is guaranteed that the output of sigmoid activation function will always be between 0 and 1, representing the probability of belonging to the positive class (1 for churn, 0 for no churn).

Model architecture summary

Layer	Number of neurons	Activation function	Dropout rate
Input layer	10	None	N/A
Dense layer 1	64	ReLU	None
Dropout layer 1	N/A	N/A	30%
Dense layer 2	32	ReLU	None
Dropout layer 2	N/A	N/A	20%
Output layer	1	Sigmoid	None